University of Saskatchewan
Computer Science

CMPT 340
Final Examination

April 28, 2007

Professor: Tony Kusalik
Format: Closed-Book†

Time: 180 minutes
Total Marks: 89

Name: _____

Student Number: _____

Directions

Answer each of the following questions in the space provided in this exam booklet and hand in the booklet when you are finished. If you must continue an answer (e.g. in the extra space on the last page or on the back side of a page), make sure you clearly indicate that you have done so and where to find the continuation.

Make all written answers legible; no marks can be given for answers which cannot be decrypted. Where a discourse or discussion is called for, please be concise and precise. Do not give "extra answers"; extra answers which are incorrect may result in your being docked marks. If you find it necessary to make any assumptions to answer a question, state the assumption(s) with your answer.

Where the exam refers to the Prolog or Haskell, assume it is the variant as implemented by SICStus Prolog or HUGS, respectively, available on the the Computer Science laboratory machines.

Marks for each major question are given at the beginning of that question. There are a total of 89 marks. Budgeting an average of 1 to 2 minutes per mark, you should easily finish within the allotted time.

Good luck.

For marking use only:

A. ____ / 5	D. ____ /15	G. ____ /15
B. ____ /28	E. ____ / 8	
C. ____ /12	F. ____ / 6	

Total: ____ /89

† Closed book, except for an optional single 8.5×11-inch quick reference sheet ("cheat sheet") of the student's own compilation.

A. (5 marks)

Consider a logic program *P*, *M(P)*, and *M*. Match each of the following relationships involving *M* and *M(P)* on the left with the conditions or conclusions on the right.

$M \subseteq M(P)$	▪	▪	program has missing answers
$M(P) \subseteq M$	▪	▪	complete
$M = M(P)$	▪	▪	program has wrong answers
$M \neq M(P)$, $M(P) \subset M$	▪	▪	correct
$M \neq M(P)$, $M \subset M(P)$	▪	▪	correct and complete

B. (28 marks)

For each of the following multiple-choice[1] questions, indicate the alternative which is the best response.

1. The "best" (computer) programming language is:

(a) Perl, since it is good for rapid prototyping.

(b) Java, because it incorporates both object-oriented and procedural language constructs, and is implemented using a virtual machine for portability.

(c) Prolog, since any knowledge can be expressed in Horn-clause logic.

(d) Haskell, since it is based on lambda calculus.

(e) Any language implemented via a meta-interpreter.

(f) A language which does not suffer from the "von Neumann bottleneck".

(g) None of the above.

2. If a language is *Turing complete*, it means that the language

(a) has a definite clause grammar (DCG), and therefore can be parsed efficiently.

(b) is non-strict.

(c) can be parsed using a Turing machine.

(d) is able to express <u>any</u> computation which any machine could perform.

(e) None of the above.

3. In Prolog, which of the following is <u>not</u> a kind of term?

(a) an atom

(b) a variable

(c) a number

(d) a functor applied to a tuple of arguments (which are terms)

(e) None of the above.

4. Which of the following functions (assuming they are defined in a programming language with suitable constructs) are *referentially transparent*?

(a) the *factorial* function (defined recursively)

(b) a function that returns a number from the keyboard

(c) a function *p* that counts the number of times it is called

(d) None of the above.

---

[1] or as Peppermint Patty from the "Peanuts" comic strip would say, "mystical guess".

5. Which of the following statements regarding dataflow languages is true?

(a) They can avoid the von Neumann bottleneck — at least conceptually — due to their inherent non-determinism.

(b) They are used primarily for business applications, where data "flows" from software component to software component.

(c) They are not appropriate for applications involving parallelism; hence they tend to be used on uni-processor systems.

(d) In dataflow languages a computation is viewed as a graph, where nodes are operations and edges are input-output links.

(e) All of the above.

6. In the context of logic programming languages, a successor to Prolog (i.e. a more state-of-the-art language or type of language) is:

(a) DCGs

(b) CLP (constraint logic programming) languages

(c) Curried functions

(d) Lambda calculus

(e) SQL

(f) None of the above.

7. Consider a Prolog predicate with a usage (instantiation) pattern specified as `p( +N, ?X )`. Here the '+' preceding the `N` indicates that the first argument

(a) may be ground.

(b) must be ground.

(c) may be non-ground (e.g. a variable).

(d) must be non-ground (e,g. a variable).

(e) may be ground or non-ground.

(f) must be ground or non-ground.

(g) None of the above.

8. Which of the following is an alternate, equivalent representation for `1:[2,3]` in Haskell?:

(a) `[1,2,3]`

(b) `1:2:3:[]`

(c) `1:(2:(3:[]))`

(d) `1:2:[3]`

(e) All of the above.

9. The main reason that interpreted languages are less efficient than compiled ones is:

(a) An optimizer stage is possible only with compilation.

(b) Interpreters suffer from a phenomenon known as the "von Neumann bottleneck" which compilers can avoid.

(c) With an interpreter, decoding of a statement must be performed at runtime and each time the statement is executed.

(d) None of the above.

10. Consider the following definition in Haskell (HUGS)

```
data Tree a = Void | Fork a (Tree a) (Tree a)
```

Here, `Void` is

(a) a type variable
(b) an overloaded type
(c) a constructor
(d) a polymorphic type
(e) a polymorphic function
(f) None of the above.

11. Variables which occur in a Prolog query

(a) are implicitly existentially quantified, i.e. read as "there exists".
(b) are implicitly universally quantified, i.e. read as "for all".
(c) are called anonymous variables.
(d) cannot appear on the left-hand side of a substitution.
(e) None of the above.

12. Which of the following classes of languages would be considered part of the larger class of *declarative languages*?

(a) logic-programming languages
(b) object-oriented languages
(c) any language which uses declarations rather than statements
(d) any language implemented via a pseudo-interpreter
(e) any language which has a high degree of writability
(f) All of the above.

13. Consider the following definition in Haskell (HUGS)

```
data Tree a = Void | Fork a (Tree a) (Tree a)
```

Here, `Tree` is

(a) a type variable
(b) an overloaded type
(c) a constructor
(d) a polymorphic type
(e) a polymorphic function
(f) None of the above.

14. Which of the following in an example of a control abstraction?

(a) a clause in Prolog
(b) an assignment statement in Java or C/C++
(c) a function in Haskell
(d) a subgoal involving "=/2" in Prolog
(e) All of the above.
(f) None of the above.

15. Consider the Prolog term `.(f,.(g,.(f(g),Xs)))`. This term is syntactically equivalent to which of the following?

(a) `[f,g,f(g)|Xs]`

(b) `[f|[g|[f(g)|[Xs|[]]]]]`

(c) `[f,g,f(g),Xs,[]]`

(d) `[f,g,f(g),Xs]`

(e) `[f,[g,[f(g),Xs]]]`

(f) All of the above.

16. Consider the following list comprehension in Haskell (HUGS)

```
[3.1 .. 7.0]
```

This list comprehension generates which of the following lists?

(a) `[3.1,4.1,5.1,6.1]`

(b) `[3.1,7.1]`

(c) `[3,4,5,6,7]`

(d) `[3.1,4.1,5.1,6.1,7.1]`

(e) None of the above.

17. The following is an expression (statement) which appears in a program in a particular programming language.

```
f( X, g( Y ) ) = Y
```

Which of the following statements is most likely to be true about this program expression?

(a) It is an example of lazy evaluation.

(b) It is an example of a unification attempt which will result in an occurs-check violation.

(c) It is an example of a non-strict function.

(d) None of the above.

18. Consider the following definition in Haskell (HUGS)

```
data Tree a = Void | Fork a (Tree a) (Tree a)
```

Here, a is

(a) a type variable

(b) an overloaded type

(c) a constructor

(d) a polymorphic type

(e) a polymorphic function

(f) None of the above.

19. Which of the following is the correct type signature for the function composition function, `(.)`, in Haskell?

(a) `(a->a)->(a->a)->(a->a)`

(b) `c->a->a->b->c->b`

(c) `(c->a)->(a->b)->(c->b)`

(d) `(a->b)->(c->a)->(c->b)`

(e) Any of the above.

20. A student types the following to HUGS

```
[a]
```

This is interpreter by HUGS as

(a) a list consisting of the character 'a' (followed by empty list)

(b) a list consisting of the variable a (followed by empty list)

(c) a string consisting of just the letter a

(d) a list comprehension

(e) None of the above.

21. Suppose Haskell (HUGS) function has the following type:

```
Int -> Int -> Int
```

This type specification is a short-hand form for the equivalent type signature which is

(a) `Int -> (Int -> Int)`

(b) `(Int -> Int) -> Int`

(c) `(Int, Int) -> Int`

(d) `Int -> (Int, Int)`

(e) None of the above.

22. In Haskell, an *operator* is

(a) another name for a function.

(b) a function which can be used to mean two (or more) different things, at different types. The equality operator == is an example.

(c) a function which is written in infix form.

(d) an identifier.

(e) None of the above.

23. Consider the following definition of a Haskell (HUGS) function:

```
multiply x y = x * y
```

Multiply will therefore be used, for example, as follows

```
multiply 2 4
```

What is the type signature of `multiply`?

(a) `Num a => (a, a ) -> a`

(b) `Num a => (a ->a ) -> a`

(c) `Num a => a -> (a -> a )`

(d) `Num a => a -> (a, a )`

(e) Any of the above.

24. What is the name for the way in which the end of a part of a definition is expressed using the layout of a script in Haskell, rather than an explicit symbol for the end?

(a) local definition

(b) offside rule

(c) cancellation rule

(d) lazy evaluation

(e) None of the above.

25. Which of the following is the correct type signature for the built-in `filter` function in Haskell?

(a) `a->Bool->[a]->[a]`

(b) `(a->Bool)->[a]->[a]`

(c) `a->Bool->[a]->[b]`

(d) `(a->b)->[a]->[b]`

(e) None of the above.

26. In a proof tree for a Prolog computation, leaf nodes are

(a) labeled by the clause and unifier used.

(b) ground.

(c) symbolized by □.

(d) facts (unit clauses) or Prolog built-ins.

(e) None of the above.

27. In Prolog, nondeterminism is achieved by

(a) backtracking.

(b) parallelism.

(c) concurrency.

(d) meta-interpretation.

(e) Any of the above.

28. Consider a function in Haskell for which either one of its arguments is a function, or its result is a function, or both. This function is said to be

(a) polymorphic.

(b) lazy.

(c) strict.

(d) higher order.

(e) All of the above.

C. (3+2+3+1+3 = 12 marks)

Answer each of the following questions with a short, technically precise answer.

1. How many items does the Haskell list `[2,3]` contain? How many does `[[2,3]]` contain? What is the type of `[[2,3]]`? (Remember: there are three answers to this question).

2. What are the two different ways of specifying comments in Haskell?

3. Give a type synonym which defines the type `Record` to be a tuple consisting of a string and an integer of arbitrary size.

4. In a lambda expression such as ($\lambda y.\lambda z.\ x\ z\ (y\ z)$), what is a role of the λ?

5. Consider the following behavior of HUGS:

```
Prelude> 1/0

Program error: {primDivDouble 1.0 0.0}
```

Suppose we define a Haskell function `foo` as follows:

```
foo x y = x
```

What will happen if we load this function definition into HUGS and then give the following expression:

```
foo 3 (1/0)
```

Why will that behaviour result?

D. (4+4+4+3 = 15 marks)

Consider a Haskell function `doubleAll` with type

```
[Float] -> [Float]
```

which doubles the value of each element of the list which is given as its (input) argument.

1. Define `doubleAll` using primitive recursion (i.e. as a basic or normal recursive function).

2. Define `doubleAll` using a list comprehension.

3. Define `doubleAll` using `map` and a local function `double`. Begin your definition

```
doubleAll xs =
```

i.e. do not use or give a Curried function.

4. Define `doubleAll` using `map` and a local function `double`. Further, define `doubleAll` as (being equal to) a Curried function.

E. (2+3+3 = 8 marks)

Answer each of the following questions with a short, precise, descriptive answer.

1. What does the following Haskell function compute? I.e. what task does it perform?

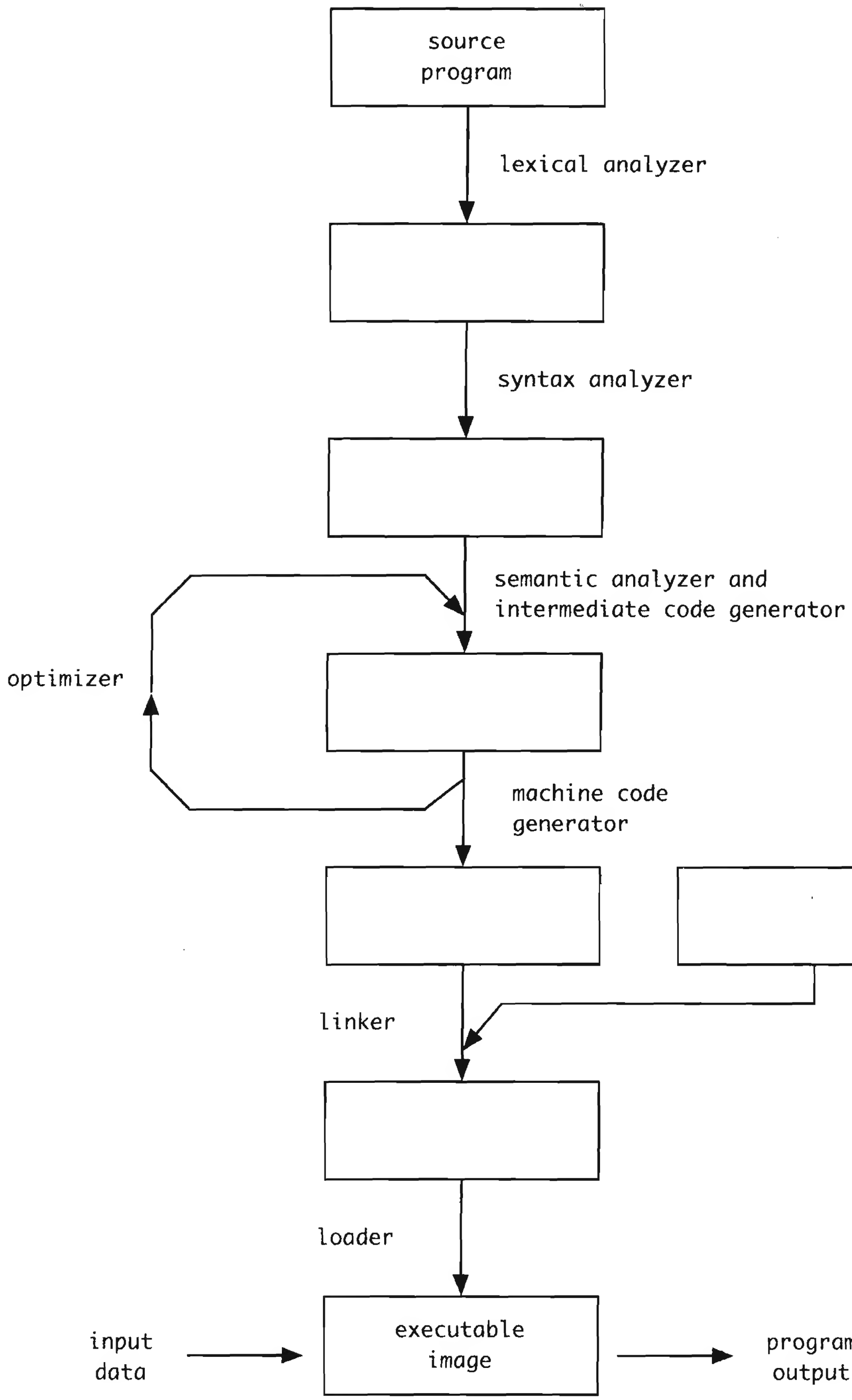
```
f [] []                 = True
f (x1:xs1) (x2:xs2)
  | x1 == x2    = f xs1 xs2
  | otherwise   = False
f _ _                   = False
```

2. What is the type of function `f` in question E.1 above? Make sure to include any type class restrictions.

3. Give a Prolog program which performs the same task as the Haskell program in E.1.

F. (6 marks)

On the next page is a diagram of the steps involved in a language implementation by way of compilation. Rectangles represent types of information, and arcs (directed line segments) represent transformations or processing stages which transform one type of information to another. Label each rectangle in the diagram with the type of information at that stage. The arcs have been already labeled for you.


source
program
lexical analyzer
syntax analyzer
semantic analyzer and
intermediate code generator
optimizer
machine code
generator
linker
loader
input
data
executable
image
program
output

G. (3+5+2+5 = 15 marks)

Answer each of the following questions with a focused, discussion-oriented answer. You are encouraged to use examples, diagrams, and other illustrative material in your responses.

1. Which of the following two formulations is more efficient? Why?

```
map ((4 +).(3*))
(map (4 +)).(map (3 *))
```

2. Distinguish between an *overloaded function* (or operator) and a *polymorphic function* (or operator) in Haskell. Illustrate your points with examples.

3. What does it mean to say that something is "treated as a first-class object". E.g., in modern functional languages, what does it mean to say that functions are treated as first-class objects?

4. Consider the various programming language design criteria discussed in class. Often, a particular programming language construct or concept will satisfy one criterion, while violating or conflicting with a second criterion. Give an example of such a situation from one programming language studied in class. Describe the construct or concept, name the two criteria, and explain how one criterion is satisfied by the construct and how the other criterion is violated.